Preparing for Automic Automation v21

Creating and Using TLS/SSL Certificates Signed by an Public CA

Version 1.4

Please note that this document is meant as a tutorial to explain TLS/SSL related concepts and how this impacts Automic Automation. This document should not be used as replacement for the product documentation.

October 25, 2022

Contents

Chapter 1: Introduction

Please note that this document is meant as a tutorial to explain TLS/SSL related concepts and how this impacts Automic Automation. This document should not be used as replacement for the product documentation.

A KeyStore is a repository of security certificates – either authorization certificates or public key certificates – plus corresponding private keys used for TLS/SSL encryption.

With the advent of Automic Automation v21, TLS/SSL is the only communication between Automation Engines, Automic Web Interface (AWI), and TLS/SSL Agents (Windows, UNIX, and Java-based Agents). As such, the implementation of certificates for secure communications is mandatory.

For production environments and Internet-facing applications, it is recommended to use certificates signed by a public Certificate Authority (CA), like IdenTrust, DigiCert, Let's Encrypt, GlobalSign, etc. These registered CAs are authorized to issue end certificates trusted by browsers and other applications using TLS/SSL to secure connections.

In this guide, we used certificates signed by Let's Encrypt and DigiCert CAs as examples for the Automic setup.

Please be aware that based on your system and version, the commands of your tools might look slightly different than in this guide.

We mention an alternative way to view certificates within this guide using the Open Source utility KeyStore Explorer. For more information, see https://keystore-explorer.org/.

.

Chapter 2: Requesting TLS/SSL Certificates From Public CAs

If your company already uses certificates signed by a public CA for other applications you can skip this step.

There are multiple ways to get certificates from public CAs and the companies behind them provide specific tools or APIs.

CA Root certificates are usually only used to sign intermediate certificates that can then verify the identities of end domains.

2.1 Let's Encrypt certificates

Let's Encrypt provides free TLS certificates and is run by the Internet Security Research Group (ISRG).

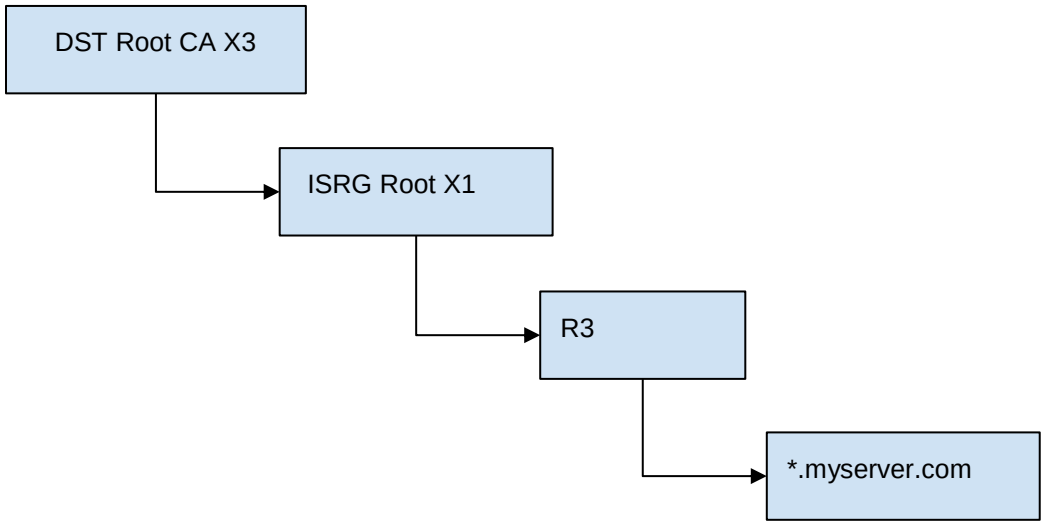
Please be aware that Let's Encrypt requires valid domain names to issue certificates and did not support IP addresses when we wrote this guide.

Your IT department should be able to help with a domain for your Automic servers.

If you require multiple end certificates, you could add them as SANs and alternatively request a wildcard certificate for a specific domain that the subdomains can use.

For example, a certificate issued for *.myserver.com can be used for both aut1-prod.myserver.com and aut2-prod.myserver.com.

The certificate chain for a wildcard domain certificate signed by Let's Encrypt could look like this:



The private key of the R3 intermediate is used to sign the server certificate. All intermediate certificates need to be configured on the Automic Server side to follow the chain of trust to the root CA certificate.

The most common method to request domain certificates from Let's Encrypt is to use the Certbot ACME client from a shell. It also provides the option to automatically renew certificates before they expire.

A command using the client could look like below:

```
$ certbot certonly --manual \
   --preferred-challenges=dns \
   --email myemail@myemail.com \
```

```
-d "*.myserver.com"
```

When using Certbot, you will get the private key matching the certificate, the certificate itself and the intermediate certificates in PEM format.

We recommend using a password to protect the private key; the default used by the Automic server is changeit.

```
/etc/letsencrypt/live/myserver.com:

privkey.pem
fullchain.pem
cert.pem
chain.pem
```

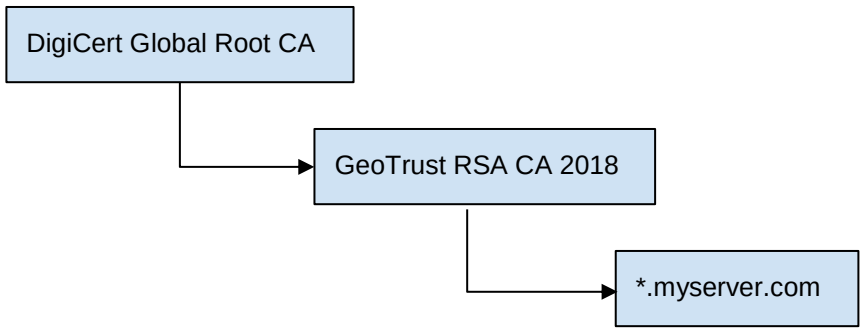
More details about the generated certificates can be found here https://certbot.eff.org/docs/using.html#where-are-my-certificates.

2.2 DigiCert certificates

DigiCert is a commercial Certificate Authority that provides TLS/SSL certificates based on subscriptions.

You can request a certificate for your domain(s) from the DigiCert website and need to provide a Certificate Signing Request (CSR) including the domain names for which the certificate should be issued.

The certificate chain for a wild-card domain certificate signed by DigiCert could look like this:



DigiCert offers an OpenSSL-based wizard that you can use to generate the command needed to create the private key and the CSR before submitting your order. The command could look like this:

```
$ openssl req -new -newkey rsa:2048 -nodes -out myserver.csr \
  -keyout privkey.pem \
  -subj "/C=AT/ST=Vienna/L=Vie/O=Broadcom/OU=Automic/CN=*.myserver.com"
```

We recommend storing the private key in a secure location and using a password to protect it. The default password used by the Automic server is `changeit`.

High Availability, Multiple Network Cards or JCPs on Multiple Servers

Suppose you have multiple network cards on servers, different internal and external addresses, or numerous JCPs across different servers. In that case, you will need to define each server domain name in the certificate.

This is done by adding Subject Alternative Names (SAN) into the CSR.

However, DigiCert recommends including only the most important server name as the Common Name in the CSR, and specifying the other domains during the order process.

Remember: If you define Subject Alternative Name, you must repeat the Common Name in the Subject Alternative Names list. This is because the Common Name will be ignored when Subject Alternative Names are present.

Chapter 3: Deploying the TLS/SSL KeyStore and Configuring the Automation Engine

3.1 Encrypting Passwords for KeyStore Configuration Settings

As a system administrator, you define the password for KeyStore connections in the **[TLS]** section of the INI files of the Automation Engine.

For security reasons, make sure that the password is encoded. To do so, you can use UCYBCRYP.EXE, which is located in the **Tools > encrypt** folder of your installation directory.

NOTE UCYBCRYP.EXE requires the C++ 2010 Redistributable Package.

Use the following parameters to enter the program via the command line:

```
UCYBCRYP[.EXE] -p -n Password
```

NOTE The password length is limited to 32 characters. The PASSWORD.UCC file, which contains the encoded password, is created in the same directory. You can copy it to the relevant INI file.

Example

```
ucybcryp -p -n uc4
```

ATTENTION An encrypted password starts with two leading hyphens. If the content of the PASSWORD.UCC file is output with the command TYPE in Windows, two exclamation marks are displayed instead of the leading hyphens; therefore, make sure to copy the password from the file.

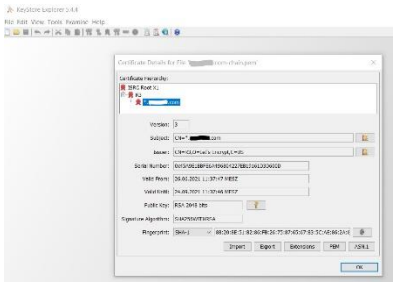
3.2 Configuring the AE INI File (ucrsv.ini) for the Automation Engine Server

The server certificate signed by the public CA (`fullchain.pem`) needs to be converted into the PKCS#12 format for the Automic server (Automation Engine) to be able to use it. The alias of the entry in the keystore is set via the name parameter and the default used by the Automic Server is `jetty`.

You will need to enter the passphrase for the key when prompted (this was set in a previous step) and set one for the keystore (for example, `changeit`).

```
$ openssl pkcs12 -export -out myserver_keystore -inkey
privkey.pem -in fullchain.pem -name jetty
```

NOTE The certificate can also be opened with the KeyStore Explorer and the information relevant for the server certificate, including the root certificate of the signing CA, can be viewed via the UI.



The final TLS/SSL configuration (using encrypted passwords) looks like this:

```
[TLS]
;
; keystore: Path and file where the keystore for TLS certificate is stored
;
```

```
keystore=myserver_keystore
;
; keystorePassword: Password of the keystore File
;
keystorePassword=¬¬103B02A4E9656774333BD58DBFE5857D33
;
; keyPassword: Password for the Keys protection
;
keyPassword=¬¬103B02A4E965677433B1B5F761168102C7;
; keyAlias: The name which the key is identified with.
;
keyAlias=jetty
```

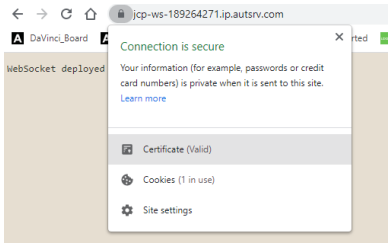
3.3 Deploying the KeyStore File to the Target Server

If you created the KeyStore on a different server you will have to copy the KeyStore file to the file identified in the AE INI file (ucsrv.ini).

3.4 Starting the Automation Engine and Testing Connections

Once all processes have started, test that the JCP endpoint is reachable.

1. In a browser, enter the server name and port number defined in the AE INI file (ucsrv.ini).
2. Make sure you check all hostnames and IP addresses configured in the certificates.
3. https://MyServer:8443/
   https://198.51.100.2:8443/
4. Open the certificate from the top left corner and check that it matches the one you previously created.

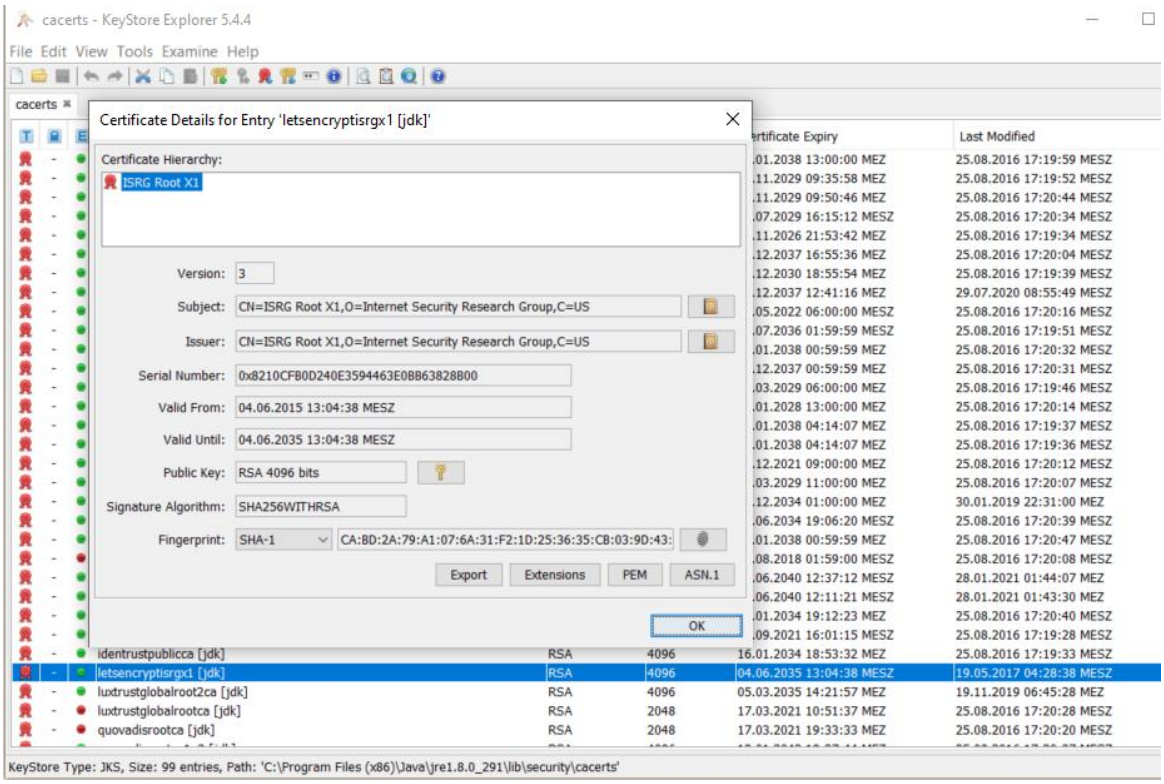


Chapter 4: Configuring the Automic Web Interface (AWI) for Public CA-Signed Certificates

For public CA-signed certificates, it is possible to establish TLS/SSL connections between AWI and the Automic Server by ensuring that AWI trusts the Certificate Authority who signed the Automic server certificate.

Browsers and other applications trust most public CAs and their root CA certificates are already deployed on all systems.

To ensure that the CA signing your certificate can be trusted by AWI, you can use KeyStore Explorer to check that the public CA root certificate (ISRG for Let's Encrypt) is included in the Java certificates truststore (cacerts).



Chapter 5: Configuring TLS/SSL Agents with Public CA-Signed Certificates

There are 2 options to use Public CA-signed certificates for secure connections between the Automic Server and the agents:

- Copy and configure all agents to use the CA-signed server certificate
- Check that the public CA root certificate is already available in the Java or OS truststores

5.1 Configuring TLS/SSL Agents to use the Certificates

You can configure the path to the public CA-Signed certificate in the INI file of every TLS/SSL Agent (Windows, UNIX, and Java-based Agents).

To do so, define the path in the **trustedCertFolder=** parameter of the **[AUTHORIZATION]** section of the relevant Agent INI file. For example:

```
[AUTHORIZATION]
trustedCertFolder = C:\AutomicAutomation\Certs
```

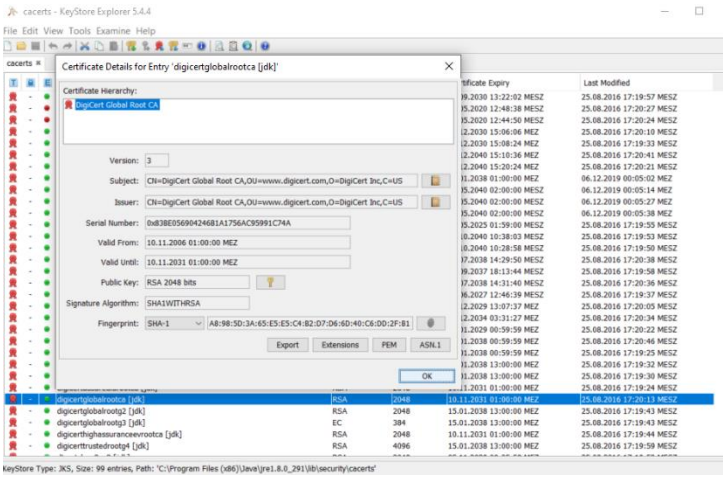
You can then deploy the exported certificates to the directory you define here.

Instead of maintaining certificates in the Automic directory structure, you can directly make use of the existing trust between certificates signed by public CAs and their root certificates already widely available in the Java, Windows or Unix truststores. In this case, you can skip configuring the trustedCertFolder parameter as described above.

5.2 Using Java TrustStore / OS Stores for Certificates

5.2.1 Java-Based Agents & TLS Gateway

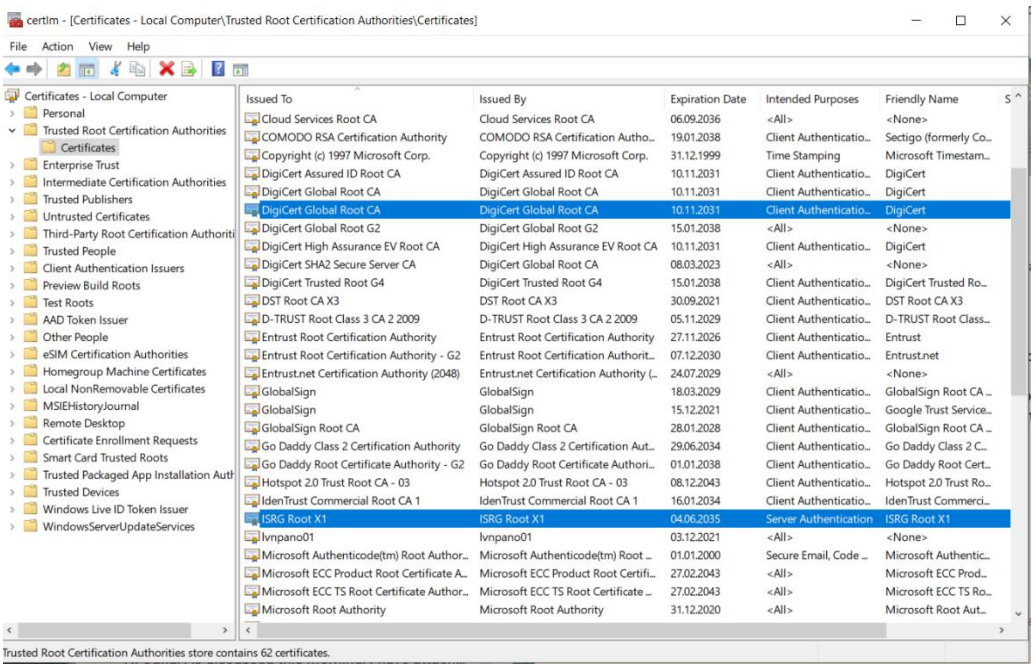
Similar to AWI, in order to make sure that the public CA signing your certificate can be trusted by the agents (RA, SAP, ...) or TLS Gateway, you can use KeyStore Explorer to check that the public CA root certificate (for example DigiCert) is included in the Java certificates truststore (cacerts).



5.2.2 Windows Agents

To verify that the Windows agents automatically trust the server when establishing TLS/SSL connections, you can use certmgr to find the root certificate of the public CA used to sign the server certificate.

For example, the root certificates for DigiCert and Let's Encrypt are already present in the Windows Trusted Root Certification Authorities.



5.2.3 Unix Agents

Since there is no standard location for the CA-signed certificates on Unix platforms, the Unix agent uses some of the most common default paths to look for trusted certificates depending on the distribution and version.

These include:

```
/etc/ssl/certs, /etc/ssl/certs/ca-certificates.crt on Debian/Ubuntu
/etc/pki/tls/certs, /etc/pki/tls/certs/ca-bundle.crt on Fedora/RHEL/CentOS
```

When the agent is started, if the sslCertDir were not set in the ini file, it will use the certificates from the above default paths if they exist.

Since the certificates can be either included as separate pem files or concatenated into one .crt file, finding the location of the trusted certificates can be a challenging task.

If openssl is installed on the system, you can determine the location of the trusted certificates folder is using the command below:

```
$ openssl version -d
```

This points to the default OPENSSLDIR where the CA certificates are installed.

But openssl is not necessarily installed on all hosts, since the Unix agent includes the libraries it needs for the TLS/SSL connections. In this case, you can search your filesystem for public CA that signed the server certificate (you might need to run the commands as sudo depending on your user's permissions).

```
$ find /etc/ssl/ -name '*DigiCert*'
/etc/ssl/certs/DigiCert_Global_Root_CA.pem

$ grep -R "DigiCert" /etc/ssl/

$ find /etc/ssl/ -name '*DST*'
/etc/ssl/certs/DST_Root_CA_X3.pem

$ find /etc/ssl/ -name '*ISRG*'
/etc/ssl/certs/ISRG_Root_X1.pem

$ find /etc/ssl/ -name '*.crt*'
/etc/ssl/certs/ca-certificates.crt

$ find  /etc/pki/tls -name '*.crt*'
/etc/pki/tls/certs/ca-bundle.crt

$ cat /etc/pki/tls/certs/ca-bundle.crt | grep DigiCert
# DigiCert Global Root CA
```

Some other possible locations for CA certificates are:

```
/var/ssl/certs/, /var/ssl/certs/ca-bundle.crt on AIX
/etc/certs/CA/, /etc/certs/ca-certificates.crt on Solaris
/etc/ssl/certs, /etc/ssl/ca-bundle.pem on SuSE Linux
```

Once you find the path where the CA root certificates are installed, if it does not match one of the defaults used by the Unix agent you need to configure it accordingly in the agent ini file (in SSLCertDir or SSLCertFile):

```
[AUTHORIZATION]
SSLCertDir = /etc/certs/CA
SSLCertFile =
```


BROADCOM®